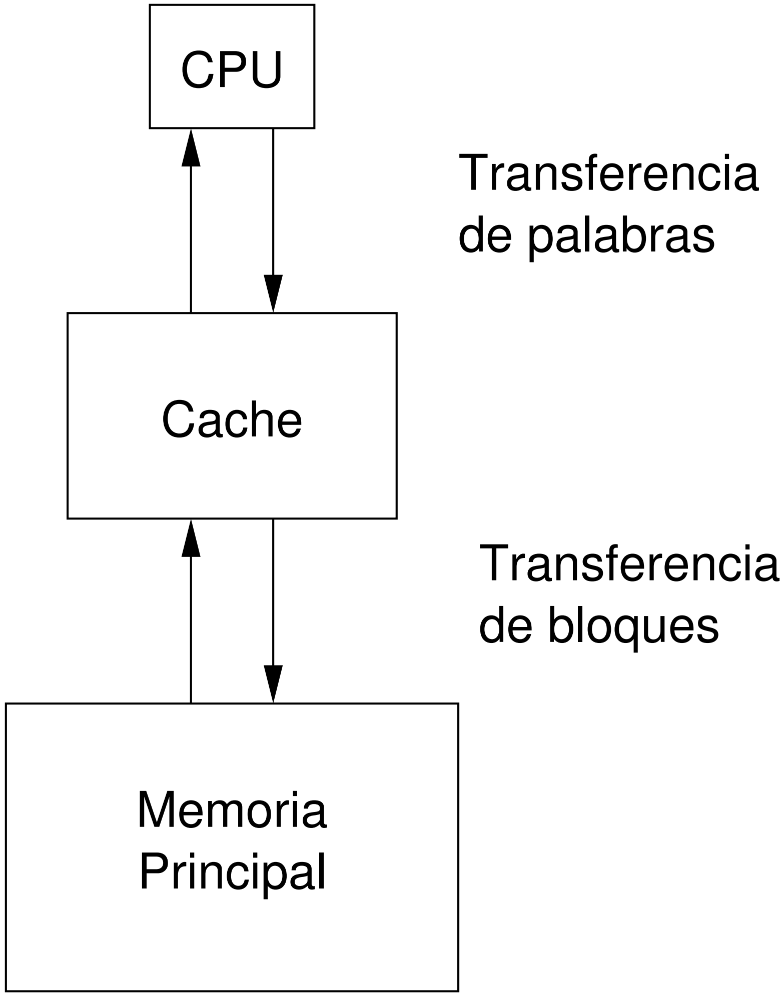
Memoria Cache

Conceptos básicos

El objetivo es aproximar la velocidad de la memoria a las de aquellas más rápidas disponibles y permitir un tamaño grande de memoria al precio de memorias semiconductoras menos costosas al mismo tiempo.

La cache contiene una copia de partes de la memoria principal, cuando el CPU intenta leer una palabra de memoria, se comprueba si está en la cache. Si es así, se entrega dicha palabra al CPU, si no, un bloque de memoria principal es transferido al cache y después la palabra se le da al CPU. Por el fenómeno de localidad de las referencias, cuando un bloque de datos es capturado, es probable que se hagan referencias futuras a otras palabras del mismo bloque.

Memorias cache y principal

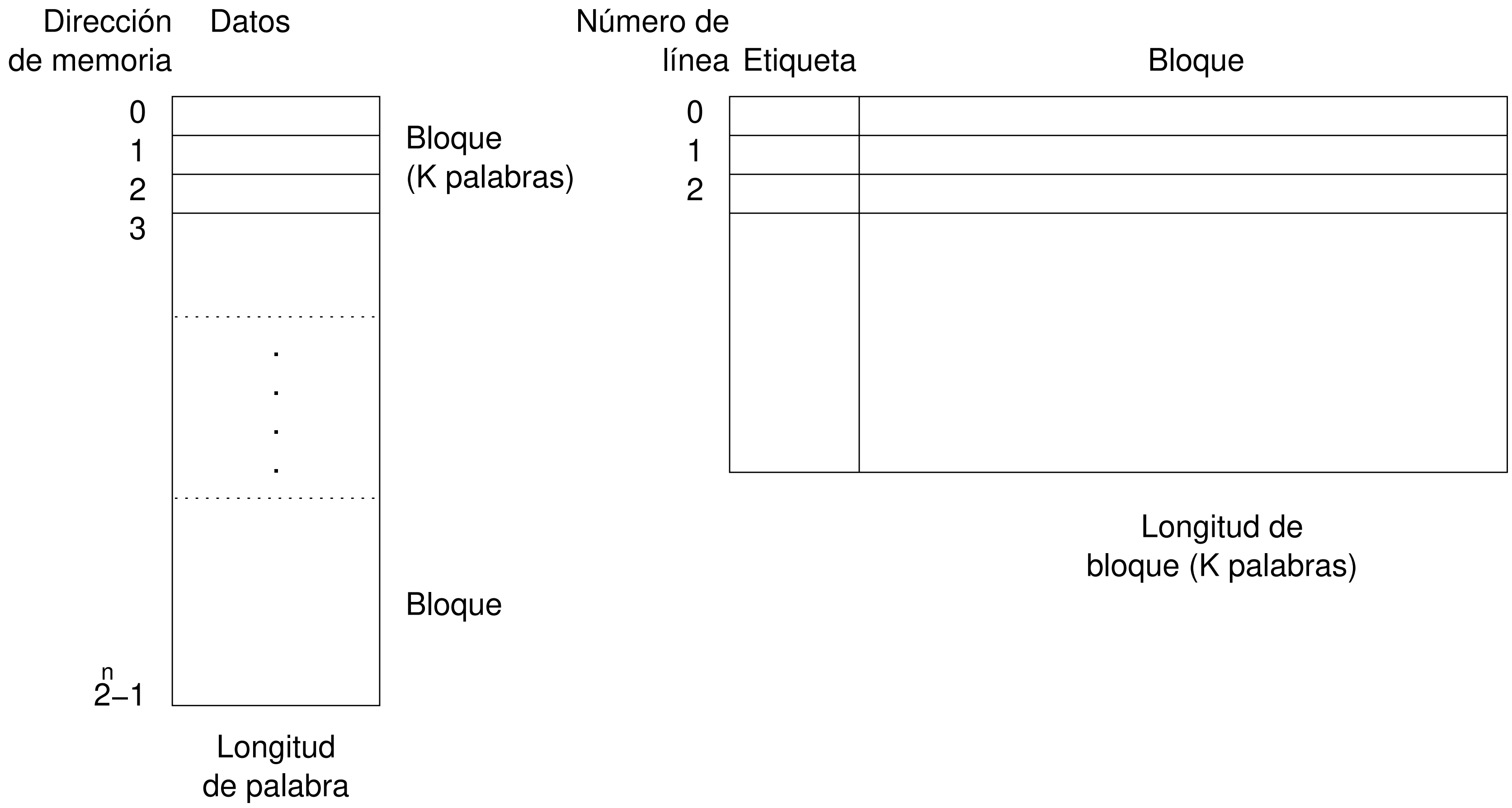


Estructura de Memoria Principal/Cache

La memoria principal consta de hasta 2^n palabras direccionables, donde cada palabra tiene una dirección única de n bits. Esta memoria esta dividida en un número de bloques de longitud fija, de K palabras por bloque, por lo tanto hay $M = 2^n/K$ bloques.

La cache consta de C líneas de K palabras cada una, donde $C \ll M$. Cada línea incluye una etiqueta que identifica qué bloque particular está siendo almacenado. La etiqueta por lo general es una parte de la dirección de la memoria principal.

Estructura de Memoria Principal/Cache



Elementos de diseño

- **Tamaño de cache** Lo ideal es que fuera lo suficientemente pequeña como para que el costo total medio por bit se aproxime al de la memoria principal, y lo suficientemente grande como para que el tiempo de acceso medio total sea próximo al de la cache. A un mayor tamaño, es mayor el número de puertas implicadas en direccionar la cache, por lo que caches grandes tienden a ser ligeramente más lentas que las pequeñas. El tamaño también se limita por la superficie disponible del chip y la tarjeta. Diversos estudio han sugerido que los tamaños óptimos se encuentran entre 1K y 512K, aunque las prestaciones del cache son muy sensibles al tipo de tarea.

Función de correspondencia

Como hay menos líneas de cache que bloques de memoria principal, se requiere un algoritmo que haga corresponder bloques de memoria principal a líneas de cache, también una forma de determinar qué bloque de memoria principal ocupa actualmente una línea dada de cache. La función de proyección determina como se organiza la cache, se pueden emplear tres técnicas: *directa, asociativa y asociativa por conjuntos*.

En los ejemplos que se revisarán se considera lo siguiente:

1. La cache puede almacenar 64 Kbytes.
2. Se transfieren los datos en bloques de 4 bytes entre la memoria y el cache, así la cache se organiza en $16K = 2^{14}$ líneas de 4 bytes c/u.
3. La memoria principal tiene 16 Mbytes, cada byte se accesa con una dirección de 24 bits ($2^{24} = 16Mbytes$).

Correspondencia directa

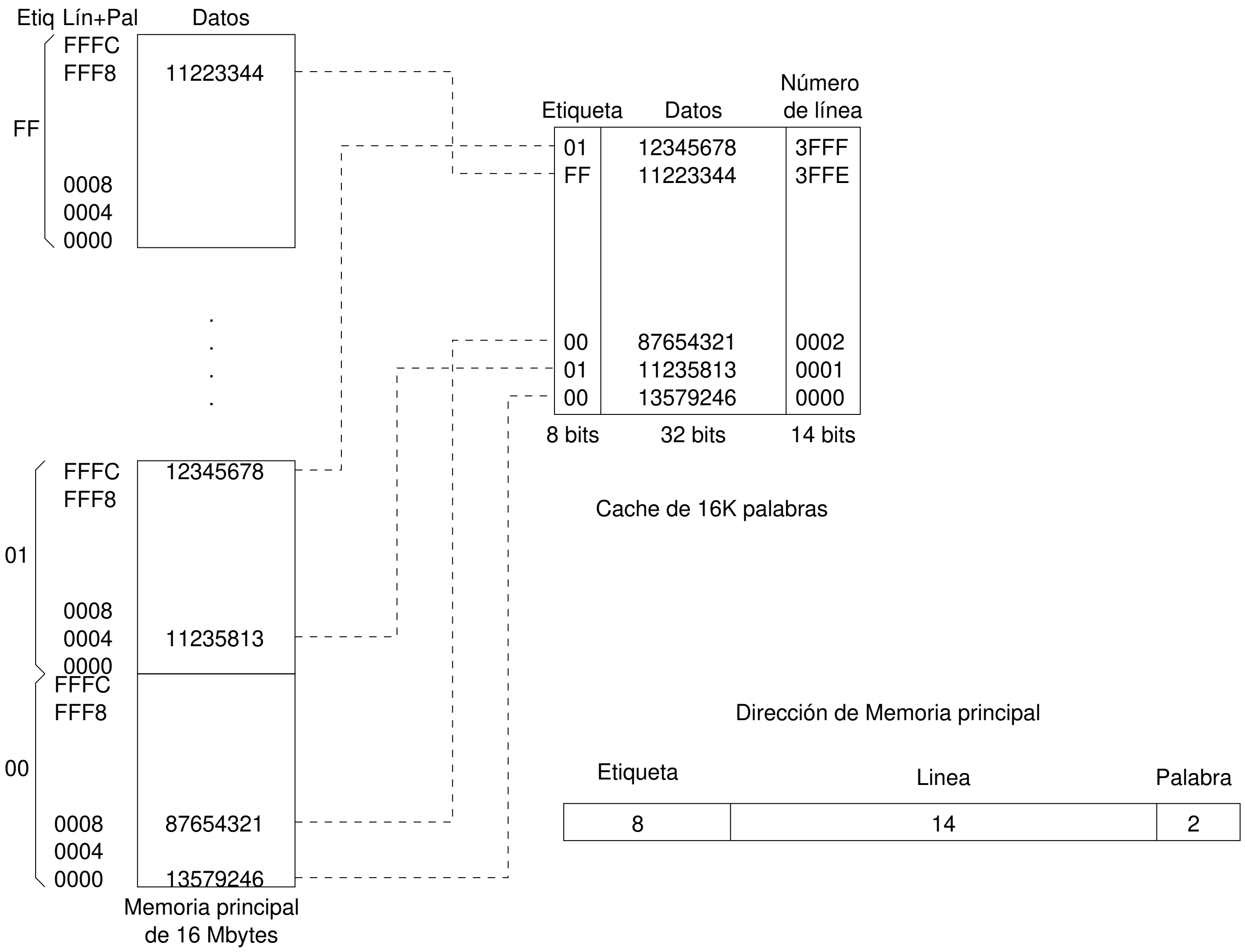
Se hace corresponder cada bloque de memoria principal a sólo una línea posible de cache. La correspondencia es:
$i = j \bmod m$
donde i es el número de línea de cache, j es el número de bloque de memoria principal y m es el número de líneas en la cache.

Los w bits menos significativos identifican cada palabra dentro de un bloque de memoria principal. Los s bits restantes indican uno de los 2^s bloques de la memoria principal. La lógica de la cache interpreta los s bits como una etiqueta de $s - r$ bits (parte más significativa) y un campo de línea de r bits, con el que se identifica una de las $m = 2^r$ líneas de la cache. Con el uso de una parte de la dirección como número de línea proporciona una asignación única de cada bloque de memoria principal en la cache.

Correspondencia de bloques de memoria principal y líneas de cache

Línea de cache	Bloque de memoria
0	$0, m, \ldots, 2^s - m$
1	$1, m+1, \ldots, 2^s - m + 1$
• • •	• • •
$m-1$	$m-1, 2m-1, \ldots, 2^s - 1$

Ejemplo de Correspondencia Directa



Para una operación de lectura se le presenta al cache una dirección de 24 bits, en donde los primeros 14 bits menos significativos son empleados como un índice para acceder a una línea particular dentro de la cache. Si el número de etiqueta, de 8 bits, coincide con el número de etiqueta almacenado actualmente en esa línea, el número de palabra de 2 bits se utiliza para seleccionar uno de los cuatro bytes esa línea. Si no, el campo de 22 bits de etiqueta línea se emplea para captar un bloque de memoria principal.

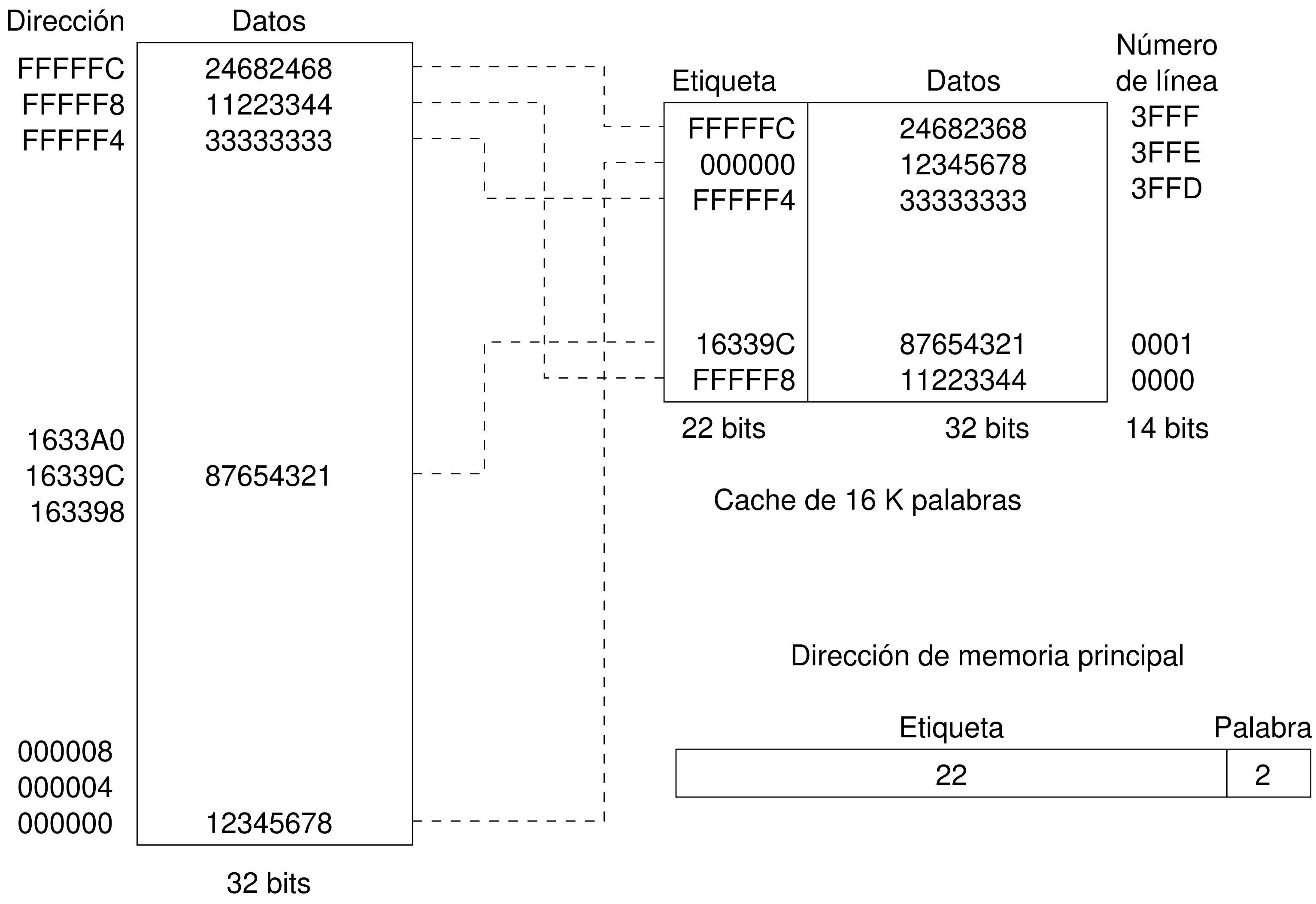
La técnica es simple y poco costosa de implementar. Su principal desventaja es que hay una posición concreta de cache para cada bloque dado. Si un programa referencia repetidas veces a palabras de dos bloques diferentes asignados en la misma línea, dichos bloques se estarían intercambiando continuamente en la cache.

Correspondencia Asociativa

Supera la desventaja de la directa al permitir que cada bloque de memoria principal pueda cargarse en cualquier línea de la cache. La lógica de control de la cache interpreta una dirección de memoria como una etiqueta y un campo de palabra. La etiqueta identifica unívocamete un bloque de memoria principal. Para determinar si un bloque está en la cache, su lógica de control debe examinar simultáneamente todas las etiquetas de líneas para buscar una coincidencia.

La principal desventaja es que se requiere una circuiteria compleja para examinar en paralelo las etiquetas de todas las líneas de cache.

Ejemplo de Correspondencia asociativa



Correspondencia asociativa por conjuntos

Es una solución de compromiso tomando lo bueno de las correspondencias anteriores, sin presentar sus desventajas. La cache se divide en v conjuntos cada uno de K líneas. Las relaciones son las siguientes:

$m = v \times k$

$i = j \bmod v$

donde i es el número del conjunto cache, j es el número de bloque de memoria principal y m es el número de líneas de la cache.

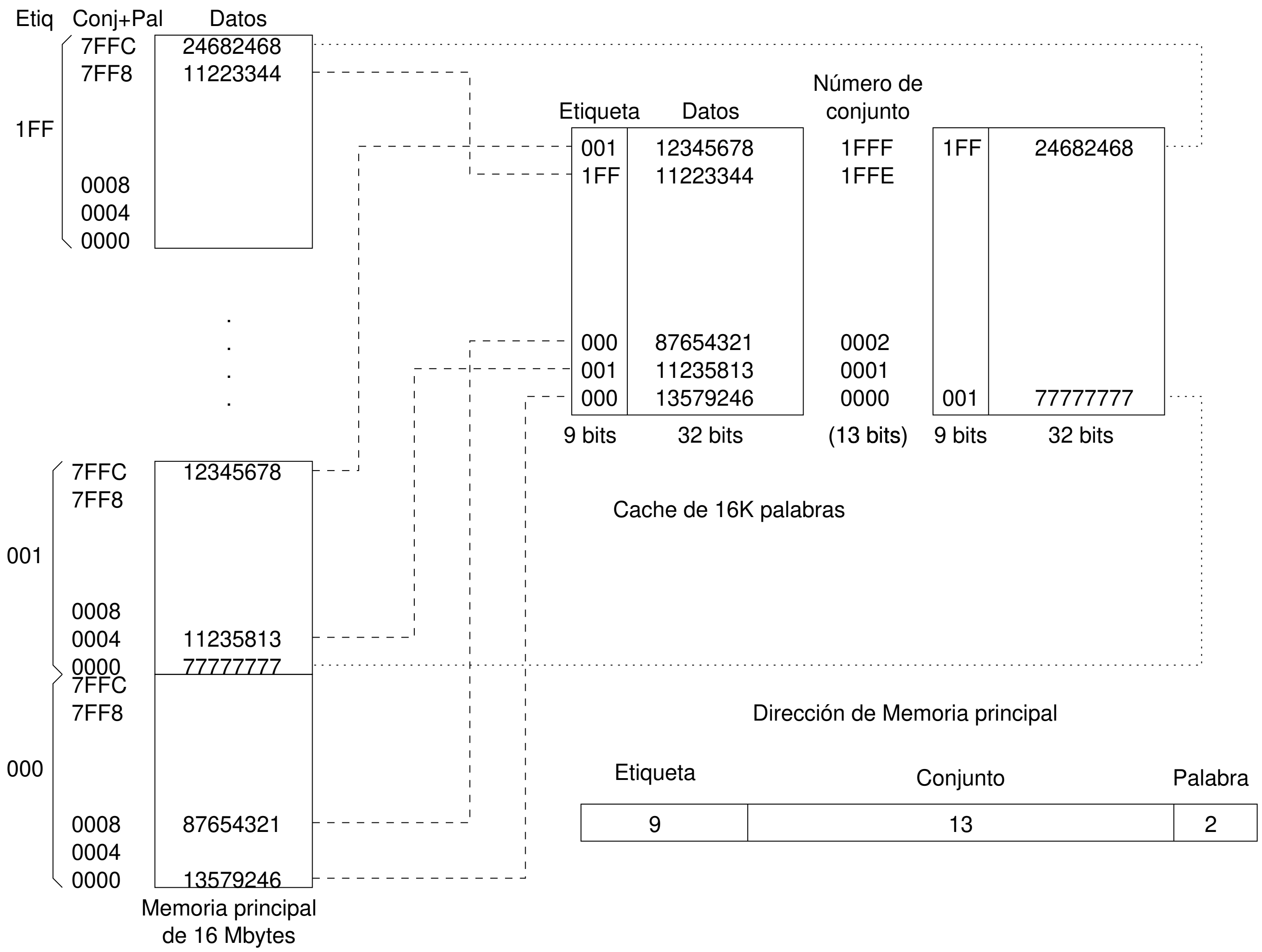
Un bloque B_j puede proyectarse en cualquiera de las líneas del conjunto i. La lógica de control de la cache interpreta una dirección de memoria con la etiqueta, el conjunto y la palabra. Los d bits del conjunto especifica uno de entre los $v = 2^d$ conjuntos. Los s bits de los campos de etiqueta y de conjunto especifican uno de los 2^s bloques de memoria principal.

Para el ejemplo tratado se tiene correspondencia asociativa por conjuntos con 2 líneas por cada conjunto. El número de conjunto, de 13 bits, identifica un único conjunto de dos líneas dentro de la cache. También da el número del bloque de memoria principal que se usa para determinar la asignación de bloques en líneas. No hay dos bloques que se hagan corresponder al mismo conjunto de la cache que tengan el mismo número de etiqueta.

Para una operación de lectura, el número de conjunto, de 13 bits, se utiliza para determinar qué conjunto de dos líneas va a examinarse. Ambas líneas del conjunto se examinan buscando una coincidencia con el número de etiqueta de la dirección a la que se va a acceder.

En el caso extremo de $v = m, k = 1$, se reduce a la correspondencia directa, y para $v = 1, k = m$ se reduce a la totalmente asociativa. El uso de dos línes por conjunto ($v = m/2, k = 2$) es el caso más común.

Ejemplo de Correspondencia Asociativa de Dos Vías



Algoritmos de sustitución

Cuando se introduce un nuevo bloque en la cache, debe sustituirse uno de los bloques existentes. Con correspondencia directa, sólo hay una posible línea para cada bloque particular y no hay elección posible, para las otras se requieren algoritmos de sustitución que sean veloces, lo que se consigue al implementarlos en hardware. El mejor es el **utilizado menos recientemente** (LRU, "Least-Recently Used"), donde el bloque que se sustituye es el que ha estado en la cache por más tiempo sin haber sido referenciado, se puede implementar en conjuntos de dos vías con un bit de uso. Otra posibilidad es el **primero en entrar – primero en salir** (FIFO, "First-In-First-Out"), se puede realiza mediante una técnica cíclica. Otra es la del **utilizado menos frecuentemente** (LFU, "Least-Frequently Used") donde un bloque se sustituye si es el que tiene menos referencias, se puede implementar asociando un contador a cada línea. Otra es usar una técnica **aleatoria**.

Política de escritura

Cuando se reemplaza un bloque que está en la cache, es necesario considerar si ha sido o no alterado en cache pero no en memoria principal. Si no lo ha sido, puede escribirse sobre el bloque antiguo de cache. Si ha sido modificado, la memoria principal debe actualizarse. Problemas:

1. Más de un dispositivo puede tener acceso a la memoria principal. Si una palabra ha sido modificada sólo en la cache, la palabra de memoria no es válida. Si un dispositivo ha alterado la memoria principal, la palabra del cache no es válida.

2. Varias CPUs conectan al mismo bus y c/u tiene su propia cache local. Si se modifica una palabra en una de las caches, podría invalidar una palabra de las otras caches.

Técnicas de escritura

- **Escritura inmediata** Todas las operaciones de escritura se hacen en cache como en memoria principal. Se genera un tráfico a memoria sustancial.
- **Postescritura** Las actualizaciones se hacen sólo en la cache. En un bloque se activa un bit asociado a cada línea cada que se escribe, cuando el bloque es sustituido, es (post-)escrito en memoria principal si el bit está activo

Si en un bus más de un dispositivo tiene una cache, y la memoria principal es compartida, se presenta un problema si se modifican los datos de una cache, ya que se inválida la palabra en memoria y en las otras caches (cuando tienen la misma palabra). Un sistema que evita lo anterior se dice que mantiene la coherencia de cache.

Aproximaciones para la coherencia de cache

- *Vigilancia del bus en escritura inmediata* Cada controlador de cache monitoriza las líneas de direcciones para detectar operaciones de escritura. Si otro maestro escribe, el controlador de cache inválida el elemento de la cache.
- *Transparencia de hardware* Con hardware adicional se asegura que todas las actualizaciones de memoria principal, vía cache, se reflejen en todas las caches.
- *Memoria excluida de cache* Sólo una porción de memoria principal se comaparte por más de un procesador, y ésta se diseña como no transferible a cache.

Tamaño de bloque (diseño)

1. Bloques más grandes reducen el número de bloques que caben en la cache.
2. A medida que un bloque se hace más grande, cada palabra adicional está más lejos de la requerida, y por tanto es más improbable que sea necesaria a corto plazo.

Un tamaño entre 4 y 8 unidades direccionables (palabras o bytes) parece estar razonablemente próximo al óptimo.

Número de caches

- *Cache de uno o de dos niveles* Con el aumento de densidad de integración se puede tener una cache en el mismo chip del procesador **cache on chip**. Se reduce la actividad del bus externo del procesador y por tanto se reducen los tiempos de ejecución, los accesos a la cache on-chip se efectúan más rápido y además el bus está libre en los períodos de acceso. Varios estudio han demostrado que el uso de un segundo nivel de cache mejora las prestaciones.

- *Cache unificada frente a partida* Recientemente la cache se ha separado en dos: una dedicada a instrucciones y otra a datos. Una cache unificada tiene varias ventajas potenciales:

  1. Para un tamaño dado de cache, una unificada tiene una tasa de aciertos mayor que una partida al nivelarse automáticamente la carga entre captación de instrucciones y de datos.

2. Sólo se necesita diseñar e implementar una cache.

La ventaja de cache partida es que se elimina la competencia por el cache entre las instrucciones y datos. Esto es importante en diseños que cuentan con segmentación de cauce (“pipelining”) de instrucciones.

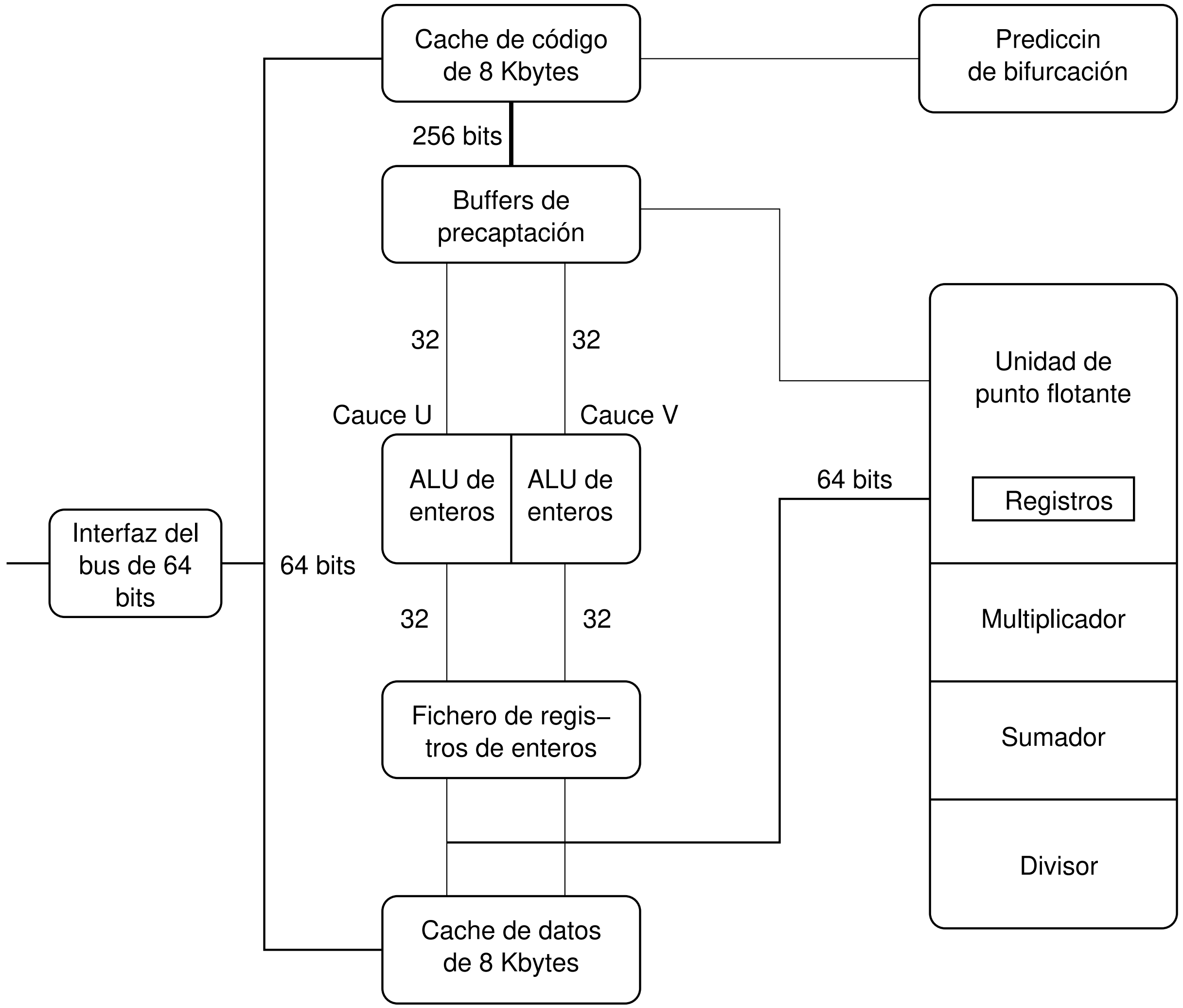
Organización de la cache del Pentium

En la evolución que se ha dado de la cache en los microprocesadores de Intel, en el 80386 no se tenía cache on chip. El 80486 incluye una sola cache on chip de 8 Kbytes, con línea de 16 bytes siendo asociativa por conjuntos de cuatro vías. En el pentium se tienen 2 caches on-chip, una para datos y otra para instrucciones, cada cache es de 8 Kbytes, con línea de 32 bytes y organización asociativa por cojuntos de dos vías.

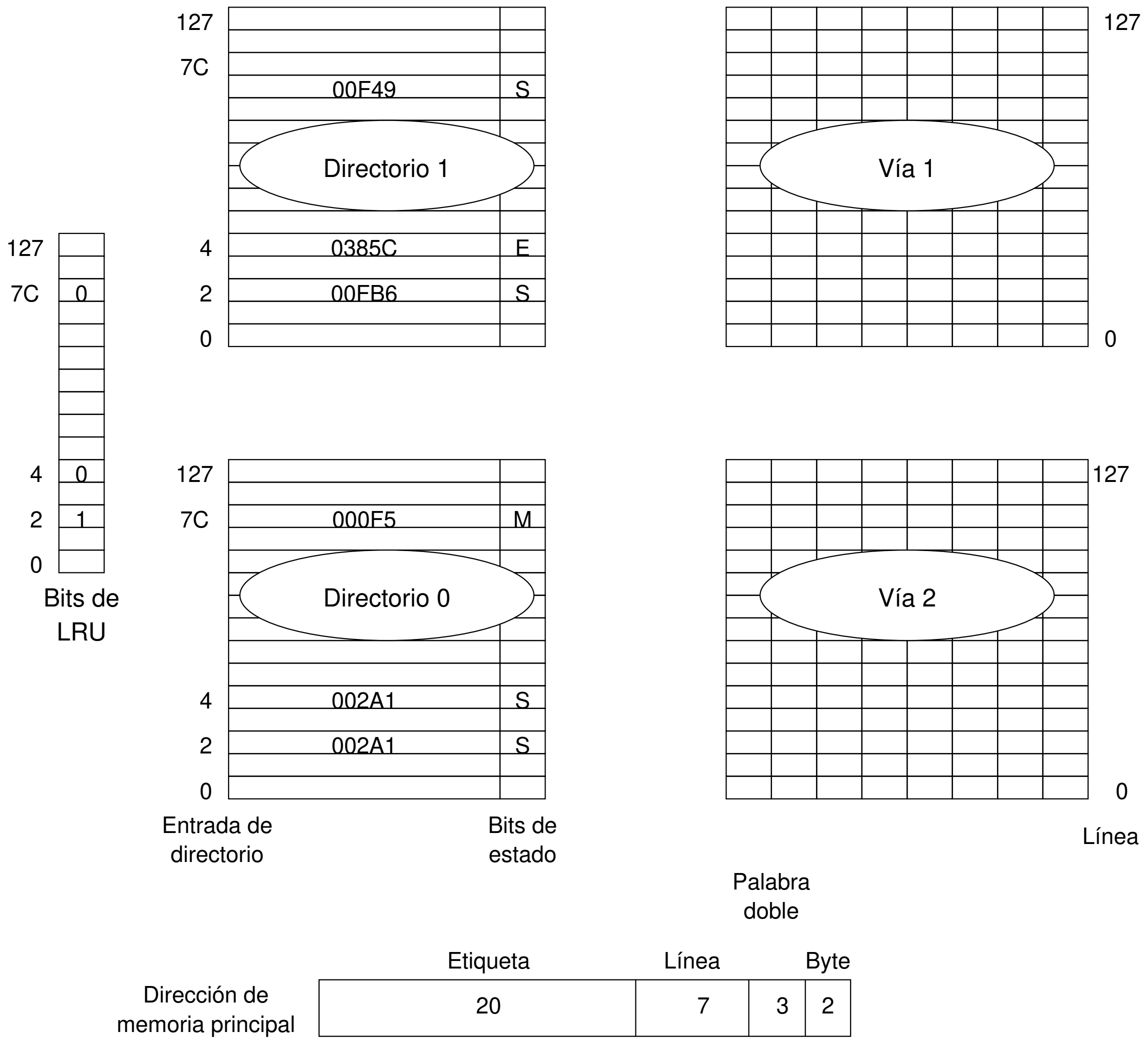
En la cache de datos se emplea la post-escritura (los datos se escriben en memoria principal cuando habiendo sido actualizados, se retiran de la cache). Se puede configurar el Pentium para utilizar la política de escritura inmediata.

El Pentium permite el uso de una cache externa de nivel 2, la cual puede ser 256 o 512 Kbytes, utilizando un tamaño de línea de 32, 64, ó 128 bytes. La cache externa es asociativa por conjuntos de dos vías.

Diagrama de bloques del procesador Pentium



Estructura de la cache interna del Pentium



Coherencia de la cache de datos

Se consigue con un protocolo denominado **MESI**(modified/exclusive/shared/invalid). Fue diseñado para satisfacer los requisitos de coherencia de cache de un sistema multiprocesador, pero también útil en arquitectura monoprocesador.

La cache de datos incluye dos bits de estado por cada etiqueta, por lo que cada línea puede estar en uno de los siguientes estados:

- *Modificada* Se modificó la línea de cache, por lo que es diferente de memoria principal y está disponible sólo en esta cache.
- *Exclusiva* Coincide la línea de la cache con lo que se tiene en memoria principal y no está presente en ninguna otra cache.
- *Compartida* Coincide con memoria principal y puede estar presente en otra cache.
- *No Válida* La línea de la cache no contiene datos válidos.

Estados MESI de una línea de cache

	M Modificada	E Exclusiva	S Compartida	I No Válida
¿Es válida esta línea de cache?	Sí	Sí	Sí	No
La copia en memoria . . .	está desfasada	es válida	es válida	—
¿Existen copias en otras caches?	No	No	Puede ser	Puede ser
Una escritura en esta esta línea . . .	no va al bus	no va al bus	va al bus y actualiza la cache	va directa- al bus

La cache interna se controla por dos bits de uno de los registros de control, **CD** (”cache disable” inhabilitar cache) y **NW** (“not write through” no escritura inmediata). También se tienen dos instrucciones que se pueden usar para controlar la cache, **INVD** para limpiar la cache e indicar que se limpie la cache externa y **WBINVD** que limpia e indica a la cache externa de post-escritura que escriba los bloques modificados antes de ser limpiada.

Organización de cache en el PowerPC

Caches internas en la familia PowerPC

Modelo	Tamaño	Bytes/Línea	Organización
601	1 32-Kb	32	Asoc por conjunto de 8 vías
603	2 8-Kb	32	Asoc por conjunto de 2 vías
604	2 16-Kb	32	Asoc por conjunto de 4 vías
620	2 32-Kb	64	Asoc por conjunto de 8 vías

En la estructura del PowerPC 620 se tienen 3 unidades aritméticas y lógicas de enteros que pueden ejecutar en paralelo, y una unidad de punto flotante con sus propios registros. La cache de datos alimenta las operaciones con enteros y las de punto flotante usando una unidad de carga/memorización. La cache de instrucciones es de sólo lectura para alimentar una unidad de instrucciones.

Las caches internas son asociativas por conjuntos de ocho vías y utilizan el protocolo MESI de para la coherencia de la cache. El protocolo se amplía para albergar un nuevo estado *Asignada* (A ”allocated). Se usa este estado cuando un bloque de datos es intercambiado y sustituido. El estado es A hasta que los datos antiguos se eescriben en el destino y se leen nuevos datos. En ese punto, el estado cambia a S o E.

Diagrama de bloques del PowerPC 620

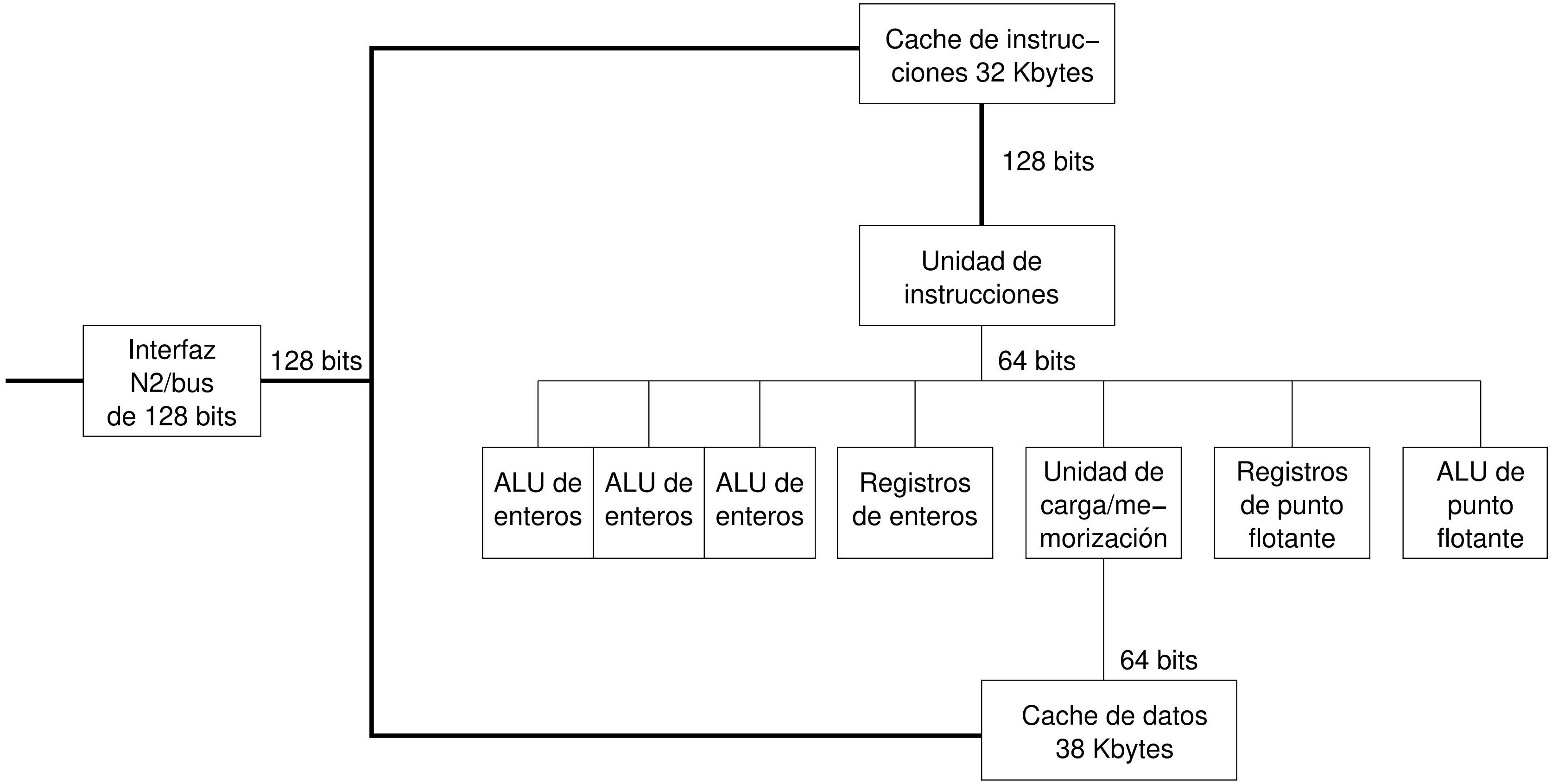
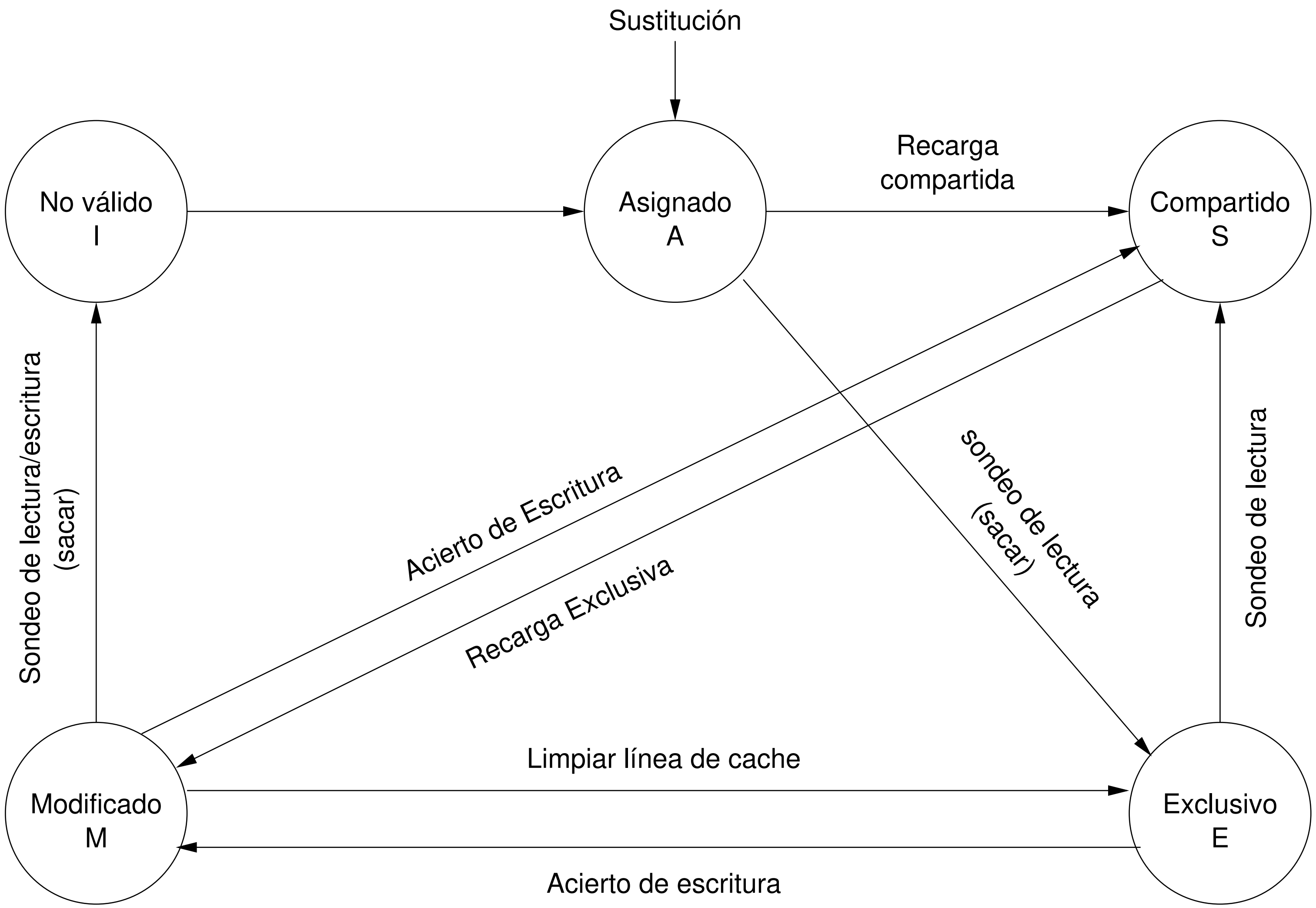


Diagrama simplificado de estados de la cache del PowerPC



Organización avanzada de la DRAM

Uno de los cuellos de botella más críticos de un sistema de altas prestaciones es la interfaz con la memoria principal interna. El bloque básico de construcción de la memoria central es el chip de DRAM. El chip DRAM tradicional está limitado por su arquitectura interna y por su interfaz con el bus de memoria del procesador.

Una forma de resolver el problema es insertar uno o más niveles de cache SRAM de alta velocidad entre la DRAM y el procesador, pero la SRAM es mucho más costosa que la DRAM, y ampliar el tamaño de cache mas allá de cierta cantidad produce menos beneficios.

Nuevas Tecnologías

- **DRAM mejorada** La EDRAM ("Enhanced DRAM") cache integra una pequeña cache SRAM en un chip de DRAM genérica. Otras características que mejoran sus prestaciones son las operaciones de refresco que pueden llevarse a cabo en paralelo con operaciones de lectura de la cache, minimizando el tiempo de no disponibilidad por causa del refresco. Como el camino de lectua desde la cache de filas es independiente del camino de escritura, se puede efectuar una operación de escritura en paralelo con un acceso de lectura a la cache.

- **DRAM cache** Es similar a la EDRAM. La CDRAM incluye una cache SRAM más grande que la anterior (16 kbits en vez de 2).

- **DRAM síncrona** La SDRAM intercambia datos con el procesador de forma sincronizada con una señal de reloj externa, que opera a la velocidad tope del bus procesador/memoria, sin imponer datos de espera.

- **DRAM Rambus** Los chips RDRAM tienen encapsulados verticales (todos los terminales en un lateral). El bus especial de las RDRAMs entrega direcciones e información del control utilizando un protocolo asíncrono orientado a bloques. Tras un tiempo de acceso inicial de 480 ns, se consigue la velocidad de datos de 500 Mbps, lo anterior se logra, ya que las RDRAMs obtienen las peticiones de memoria a través de un bus de alta velocidad. Cada petición contiene la dirección deseada, el tipo de operación, y el número de bytes en dicha operación.

- **RamLink** Se centra en la interfaz procesador/memoria en lugar de la arquitectura interna de los chips DRAM. La RAMLink es una interfaz de memoria con conexiones punto a punto dispuestas en un anillo. El tráfico en el anillo es gestionado por un controlador de memoria que envía mensajes a los chips de DRAM, los cuales actúan como nodos de la red en anillo.